Claim Amendments

1.   (Currently Amended) A method comprising:

generating a function having an argument, the function expressed in a high-level programming language, wherein the function includes a set of one or more instructions that instruct a compiler unit to return a memory address of the argument as a result of the function; and

generating a call to the function, the call expressed in the high-level programming language, wherein the call causes the compiler unit to pass a descriptor as the argument, the descriptor pointing to a target data.

2.   (Original) The method of claim 1, wherein the high-level programming language includes a Fortran programming language.

3.   (Original) The method of claim 1, wherein the argument is an integer data type.

4.   (Currently Amended) A method comprising:

receiving a first code, wherein the first code refers to a variable of a target data type, wherein the variable is addressable using a descriptor; and

translating the first code into a second code, the second code expressed in a high-level programming language, wherein the translation requires a memory address of the descriptor, and wherein the translation comprises:

generating a function having an argument, wherein the function is expressed in the high level programming language, and the function includes a set of one or more instructions that instruct a compiler unit to return the memory address of the argument as a result of the function; and

generating a call to the function, wherein the call causes the compiler unit to pass the descriptor as the argument.

5.   (Original) The method of claim 4, wherein the translating comprises generating an interface block for the function for each different target data type in the first code.

6.   (Original) The method of claim 4, wherein the high-level programming language includes a Fortran programming language.

7.   (Original) The method of claim 4, wherein the argument is an integer data type.

8.   (Original) The method of claim 4, further comprising generating a data structure to store information based on the target data type.

9.   (Original) The method of claim 7, wherein the function includes a routine from a runtime library, the routine to return a memory address of an argument of the routine.

10. (Original) The method of claim 9, wherein the routine from the runtime library is written in a C programming language.

11. (Currently Amended) A machine-readable medium that provides instructions, which when executed by a machine, causes the machine to perform operations comprising:

generating a function having an argument, the function expressed in a high-level programming language, wherein the function includes a set of one or more instructions that instruct a compiler unit to return a memory address of the argument as a result of the function; and

generating a call to the function, the call expressed in the high-level programming language, wherein the call causes the compiler unit to pass a descriptor as the argument, the descriptor pointing to a target data.

12. (Original) The machine-readable medium of claim 11, wherein the high-level programming language includes a Fortran programming language.

13. (Original) The machine-readable medium of claim 11, wherein the argument is an integer data type.

14. (Currently Amended) A machine-readable medium that provides instructions, which when executed by a machine, causes the machine to perform operations comprising:

receiving a first code, wherein the first code refers to a variable of a

target data type, wherein the variable is addressable using a descriptor; and

translating the first code into a second code, the second code

expressed in a high-level programming language, wherein the translation requires a

memory address of the descriptor, and wherein the translation comprises:

generating a function having an argument, wherein the function

is expressed in the high level programming language, and the function includes a set

of one or more instructions that instruct a compiler unit to return the memory address

of the argument as a result of the function; and

generating a call to the function, wherein the call causes the

compiler unit to pass the descriptor as the argument.

15.   (Original) The machine-readable medium of claim 14, wherein the

translating comprises generating an interface block for the function for each different

target data type in the first code.

16.   (Original) The machine-readable medium of claim 14, wherein the high-

level programming language includes a Fortran programming language.

17.   (Original) The machine-readable medium of claim 14, wherein the

argument is an integer data type.

18.   (Original) The machine-readable medium of claim 14, further comprising generating a data structure to store information based on the target data type.

19.   (Original) The machine-readable medium of claim 14, wherein the function includes a routine from a runtime library, the routine to return a memory address of an argument of the routine.

20.   (Original) The machine-readable medium of claim 19, wherein the routine from the runtime library is written in a C programming language.

21.   (Currently Amended) A system comprising:

a translation unit to receive a first code that refers to a variable of a target data type, wherein the variable is referred to by a descriptor, the translation unit to translate the first code into a second code, the second code based on a high-level programming language, wherein the translation requires a memory address of the descriptor and wherein the translation comprises:

generating a function having an argument, wherein the function is expressed in the high level programming language, and the function includes a set of one or more instructions that instruct a compiler unit to return the memory address of the argument as a result of the function; and

generating a call to the function, wherein the call causes the compiler unit to pass the descriptor as the argument;

a compiler unit coupled with the translation unit, the compiler unit to receive the second code and to generate object code, wherein the compiler unit passes the descriptor as the argument based on the call to the function-; and

a linker unit coupled with the compiler unit to link the object code and a number of routines to generate executable code, wherein the address of the descriptor can be retrieved through one of the number of routines when the executable code is executed.

22. (Original) The system of claim 21, wherein the generation of the second code includes the generation of a function, the function having an entity as an argument, and a call to the function, wherein the call to the function instructs a compiler unit to accept the argument as an entity for which the memory address can be determined and returned as a result of the function.

23. (Currently Amended) The system of claim 21 22, wherein the entity is an integer.

24. (Original) The system of claim 21, wherein the high-level programming langue includes a Fortran programming language.

25. (Original) The system of claim 21, wherein the function includes a routine from a runtime library, the routine to return a memory address of an argument of the routine.

7

26.   (Original) The system of claim 25, wherein the routine from the runtime

library is written in a C programming language